The Context Layer

Source-Aware Compression for Stateless LLM Inference in Business Workflows

Shishir Govinda M[1*]

[1]Co-Founder & CTO, Tvara, Bengaluru, India

ABSTRACT

Large Language Models (LLMs) are increasingly used as reasoning engines inside business workflows, yet model inference remains stateless at the model level: each call can only reason over the information supplied at that moment. This creates a critical limitation for enterprise AI systems, where useful business context is distributed across emails, calls, CRM records, documents, transactions, support tickets, and internal notes. This paper introduces the concept of the *Context Layer*: a system layer between raw business data and stateless LLM inference that retrieves, classifies, compresses, structures, ranks, and assembles task-specific context for each model call. The paper argues that enterprise AI performance depends not only on model capability or context-window length, but on the density, structure, provenance, recency, persistence, and relevance of the context supplied during inference. We define *context density* as the amount of task-relevant, source-grounded business signal contained per token of context, and propose that high-density structured payloads such as context cards, tables, JSON records, timelines, and selected verbatim evidence can improve reasoning compared to raw long-context injection. The framework distinguishes between tool access and context understanding: even when an agent can access CRMs, databases, files, APIs, or communication systems, it must still decide what to inspect, what to trust, what to ignore, what to compress, and what to carry forward. The paper further develops a utility-based view of context selection that balances recency with semantic importance, source reliability, persistence of commitments, redundancy, permissions, token cost, and latency. We propose an evaluation framework comparing no context, raw full context, retrieved snippets, generic summaries, structured context, source-aware context cards, and utility-weighted context cards across business tasks such as lead prioritization, next-best-action generation, deal-risk detection, follow-up drafting, CRM update recommendation, and conversation summarization. The central thesis is that business AI systems should not simply maximize how much a model can read; they should optimize what the model needs to know, in what structure, from which source, and under what operational constraints.

Keywords: Context Layer, Context Engineering, Stateless Inference, Large Language Models, Context Density, Context Compression, Source-Aware Context, Structured Context, Business AI, Enterprise AI Workflows, Retrieval-Augmented Generation, Agentic Systems

1. INTRODUCTION

Large Language Models (LLMs) are increasingly being used as reasoning engines inside business software. They summarize conversations, draft responses, recommend next actions, analyze customer intent, classify leads, extract insights from documents, and support operational decision-making. However, despite their growing role in enterprise workflows, LLMs do not naturally possess the live memory of a business. An LLM can be understood as a conditional generation system whose response is shaped by learned parameters and the context supplied during inference [1], [2]. This makes context construction central to the performance of LLM-based systems.

Every model inference call is, at the model level, stateless. The model responds based on the prompt, system instructions, retrieved context, tool outputs, memory snippets, and other information provided within that specific call. Once the call is complete, the model does not automatically retain the business state unless that state is stored externally, retrieved, and reintroduced in a later call. Modern agent frameworks may provide persistence, session memory, tool access, and workflow orchestration, but these mechanisms remain external to the model itself. This creates a foundational systems challenge: if the model is only as informed as the context supplied to it, then the quality of the AI output depends heavily on the quality of the context layer beneath it.

Most current approaches treat this challenge as a retrieval, memory, or tool-access problem. Retrieval-Augmented Generation (RAG) has shown that external knowledge can improve factual grounding

Published: 20 June 2026;
*Corresponding Author: Shishir Govinda M <s.govinda@tvara.global>

for knowledge-intensive tasks [3]. More recent systems extend this idea through tool-integrated reasoning, memory architectures, agent workflows, and protocols that allow models to interact with external systems. These systems improve access to information, but access alone does not solve the context problem. Even when an agent can access a database, CRM, email inbox, document store, or transaction system, it must still decide what to inspect, which source to trust, what to ignore, what to compress, and what to supply to the model for the current task.

This distinction is especially important in business workflows. A business decision rarely depends on all available data. It depends on the right signals, selected from the right sources, at the right time. For example, deciding the next best action for a sales lead may not require the entire CRM history, all prior emails, and every call transcript. It may require only the latest objection, the buyer's stated intent, the product discussed, the last promised follow-up, the current deal stage, and the most reliable source for each of these facts. In this setting, more context does not always mean better context.

A further challenge is that the system often needs context in order to decide what context to retrieve. To determine which database record, email thread, call transcript, or document section is relevant, the system must first know what sources exist, how they are indexed, when they were updated, and what business meaning they carry. This creates a sequential context-discovery problem: before useful context can be assembled, the system must maintain an inventory of available context sources and their metadata. Without such an index, the model may either retrieve too little information, retrieve irrelevant information, or over-rely on noisy long-context input.

This paper proposes the term *Context Layer* to describe the missing system layer between raw business data and stateless LLM inference. The Context Layer is responsible for converting fragmented business memory into compact, source-aware, task-specific context. Its objective is not to maximize the amount of context supplied to the model, but to maximize the usefulness, reliability, and business relevance of each token supplied.

The central claim of this paper is simple: in business AI workflows, the performance of a generalist LLM depends less on the total size of the context window and more on the density, structure, source, recency, and relevance of the context provided during inference. The Context Layer is therefore not merely a retrieval mechanism; it is a source-aware context planning, compression, and assembly layer for operational decision-making.

1.1. Why Tool Access Does Not Eliminate the Context Problem

Recent progress in LLM-based software development, tool calling, workflow agents, and long-running autonomous agents may suggest that the context problem can be solved by simply giving the model access to more systems. If an agent can query a database, inspect files, call APIs, execute code, search documents, and interact with a user interface, it may appear that explicit context construction is no longer necessary. However, tool access and context availability are not equivalent to context understanding.

A tool-using agent still faces a prior decision problem: before it can use the right information, it must decide which information source to inspect, which tool to call, what query to issue, which result to trust, and how much of the result should be carried forward into the next inference call. This makes context selection a sequential systems problem rather than a single retrieval problem. To know what context is required, the agent must first know what context exists. This requires an index of available sources, metadata about recency and reliability, and a representation of how different sources relate to the task [4], [5], [6], [7].

This issue becomes more visible in software engineering workflows. As more developers use LLMs to generate code, the local act of producing a function, component, or API endpoint becomes easier. However, system-level reasoning can become weaker if the model is not supplied with architectural context: existing service boundaries, database schemas, authentication flows, deployment constraints, failure modes, ownership rules, and long-term maintainability requirements. A model may generate syntactically correct code while still violating the system design. In such cases, the failure is not only a coding failure; it is a context failure.

Long-running agents further demonstrate this distinction. Autonomous agents can operate across terminals, repositories, APIs, and execution environments, but their reliability depends on persistent project instructions,

memory across sessions, explicit success criteria, and evaluable feedback loops. Files such as project instructions, changelogs, test suites, commit histories, and reference implementations act as externalized context structures. They prevent the agent from repeatedly exploring failed paths, losing progress across sessions, or optimizing for short-term completion over system correctness [8].

The same principle applies to business workflows. Giving an agent access to a CRM, email inbox, call transcript database, document store, and transaction system does not guarantee that it will assemble the right business context. It may retrieve stale records, over-weight noisy messages, ignore source reliability, or fail to distinguish between a casual comment and a contractual commitment. Therefore, access must be mediated by a context layer that maintains source inventories, classifies business meaning, tracks recency, compresses relevant information, and assembles the final context payload for inference.

This paper therefore treats tool access as one component of context engineering, not as a replacement for it. Tools expand what the system can reach; the Context Layer determines what the model should know. Without this separation, LLM-based systems risk replacing explicit system thinking with opportunistic generation. With it, agents can be made more reliable because each inference call is grounded in a compact, source-aware, and task-relevant representation of the larger system state [9], [10].

2. LITERATURE REVIEW

Research on LLM application design has increasingly shifted from isolated prompt construction toward context engineering: the systematic optimization of the information payload supplied to a model during inference. Recent surveys describe context engineering as a broader discipline that includes context retrieval, context processing, context management, RAG, memory systems, tool-integrated reasoning, and multi-agent orchestration [4].

A key motivation for context engineering is that long context is expensive and not always reliable. Transformer attention introduces computational and memory costs that grow with sequence length [2]. Work on selective context demonstrates that input redundancy can be pruned to reduce context cost while maintaining comparable downstream performance [11]. This supports the idea that context should be treated as an optimizable resource rather than a passive container.

Long-context models also face utilization challenges. Studies of the *lost in the middle* phenomenon show that models can struggle when relevant information is buried in the middle of long contexts [12]. Follow-up work attempts to strengthen attention and improve multi-document question answering in long contexts [13]. Other research shows that effective context length may fall short of the nominal context window because of positional and training-distribution effects [14]. These findings suggest that expanding the context window alone is insufficient for reliable business reasoning.

Several studies investigate how LLMs internally use context. Layer-wise probing work suggests that context knowledge is encoded differently across model layers and can be disrupted by irrelevant evidence [15]. Research on in-context learning suggests that models may transition from task recognition to task performance after certain layers, and that context attention can become partially redundant in later computation [16]. Layer-wise analyses further describe a compression-expression dynamic, where earlier layers compress task information and later layers express it toward the final output [17]. These findings align with the proposed Context Layer: the system should supply compact and task-relevant information rather than assuming that raw context will be optimally handled by the model.

Acceleration work such as self-speculative decoding and in-context layer skipping also highlights that not all computation is equally necessary for every context and task [18]. Although this paper focuses on application-layer context rather than model-internal computation, both directions point to the same principle: inference efficiency and output quality can improve when redundant processing is reduced.

3. PROBLEM STATEMENT

Business workflows are continuous, stateful processes, while LLM inference calls are discrete and stateless at the model level. A sales deal may unfold across emails, calls, meetings, WhatsApp messages, CRM updates,

proposals, pricing discussions, and internal notes. A customer issue may involve support tickets, product logs, payment records, prior complaints, escalation history, and contractual commitments. In each case, the useful context is distributed across multiple systems and changes over time.

A stateless inference call cannot reason over this distributed history unless the relevant parts are supplied to it. This creates three connected problems: a technical context problem, a business reasoning problem, and a cost-access-evaluation problem.

3.1. Technical Context Problem

The first problem is technical. Business context is often too large, fragmented, and heterogeneous to pass directly into a model. Processing thousands of emails, long call transcripts, large PDFs, CRM records, and event logs may require multiple inference calls for retrieval, extraction, summarization, classification, and final reasoning. Each additional call introduces latency, cost, and possible information loss. Batch processing can improve throughput, but it does not solve the core problem of deciding which context is relevant, reliable, recent, and necessary for the current task.

Long-context models reduce some of these constraints, but they do not eliminate them. Transformer-based models still incur significant computational cost when processing long inputs, and recent work shows that the effective use of long context can fall short of the theoretical context window [11], [13], [14]. Therefore, the ability to pass more tokens should not be treated as equivalent to the ability to reason better over business state. A 200-page PDF, 5,000 emails, or months of CRM activity may contain the answer, but the model may still fail if the relevant signal is buried inside noisy, redundant, or poorly ordered context.

This creates a context planning requirement. Before an LLM can answer a business question, the system must decide what sources to inspect, what information to extract, what to summarize, what to preserve verbatim, and what to discard. Without such planning, enterprise AI systems either under-contextualize the model, leading to shallow or hallucinated answers, or over-contextualize it, increasing cost and confusion.

3.2. Business Reasoning Problem

The second problem is business reasoning. Business decisions are not purely statistical predictions over historical data. Statistical patterns are useful as priors: they can reveal conversion rates, seasonal trends, average response times, common objections, and expected deal movement. However, an active business transaction is not interchangeable with another transaction, even if both belong to the same segment or deal size.

For example, two enterprise deals worth one million dollars may look similar in aggregate reporting, but differ in buyer motivation, internal urgency, procurement constraints, stakeholder influence, competitive pressure, and timing. A sales executive cannot rely only on the mean or median behavior of past deals when deciding what to say in the current conversation. The decision must also consider grounded, transaction-specific context: what this buyer said, what was promised, what changed recently, which objection is unresolved, who has authority, and what source confirms each fact.

This creates a distinction between aggregate analytics and transaction-level reasoning. Aggregate analytics can describe what usually happens across many deals. Transaction-level reasoning must decide what should be done for this specific deal at this specific moment. The latter requires source-aware context, because the meaning of a business signal depends on where it came from and when it occurred.

Business workflows also contain uncertainty that cannot be fully controlled or observed. A buyer's mood, organizational pressure, market timing, weather, seasonality, internal politics, or competing priorities may influence the outcome. These factors make business outcomes inherently uncertain, even when the system has strong historical data. Therefore, enterprise AI should not present recommendations as deterministic truths. It should produce grounded, context-aware recommendations with explicit assumptions, confidence levels, and traceable supporting evidence.

3.3. Cost, Access, and Evaluation Problem

The third problem involves cost, access, and evaluation. In real enterprise systems, not every piece of context is equally available or equally appropriate to use. Some data may be private, permission-restricted, stale, duplicated, incomplete, or stored in systems that are expensive to query. An agent may have access to a CRM, email inbox, document store, and transaction database, but access alone does not determine what should be used. Context must be filtered through permissions, source reliability, recency, and task relevance.

Evaluation is also difficult in business workflows. Unlike benchmark tasks with fixed correct answers, business outcomes are affected by many external variables. If an AI system recommends a follow-up message and the deal closes, the outcome may be due to the recommendation, the salesperson's relationship, timing, buyer budget, competitor weakness, or unrelated internal changes. Conversely, a good recommendation may fail because of factors outside the system's control. This makes direct evaluation of business AI outputs harder than ordinary classification or question-answering tasks.

For this reason, evaluating a Context Layer requires multiple levels of measurement. At the context level, the system should measure relevance, compression quality, source coverage, and freshness. At the output level, it should measure groundedness, source attribution, hallucination rate, actionability, and consistency. At the workflow level, it should measure latency, token cost, human acceptance, downstream conversion, and error recovery. No single metric is sufficient.

These challenges suggest that enterprise AI systems require a dedicated Context Layer. Without such a layer, the model is forced to reason over either insufficient context or excessive context. Both cases reduce output quality. The central problem is therefore not only how to retrieve more information, but how to transform fragmented business memory into compact, source-aware, task-specific context for each inference call.

4. WHY RAW CONTEXT IS NOT ENOUGH

A common assumption in LLM application design is that better outputs can be achieved by giving the model more information. While additional context can improve performance in some cases, it fails as a general principle for business workflows. The problem is not only the absence of information, but the absence of usable information.

Raw context is often large, noisy, and repetitive. A 200-page PDF may contain only a few clauses relevant to a pricing decision. A list of 100 emails may include repeated follow-ups, confirmations, outdated attachments, forwarded chains, and low-value social exchanges. A call transcript may include greetings, filler speech, transcription errors, interruptions, and unrelated discussion. Passing all of this information into the model increases token load, latency, and cost without necessarily increasing decision quality. Prior work on context compression shows that reducing redundant input can significantly improve inference efficiency while preserving comparable output quality [11].

Raw context also introduces attention and retrieval risk. Even when a model supports a long context window, there is no guarantee that it will use the most relevant evidence effectively. Long-context studies show that models may struggle when important information is buried inside large context blocks, and that effective context use can fall short of the theoretical context length [12], [13], [14]. In business workflows, this means that simply appending more emails, documents, transcripts, or CRM notes may make the model appear better informed while actually making the decision process less reliable.

Raw context also lacks business structure. A model may receive a large amount of text but not know which parts represent commitments, objections, requirements, preferences, risks, assumptions, or outdated statements. In operational workflows, the meaning of information depends on its role in the business process. For example, the sentence "We are not ready this month" could mean different things depending on the source and stage. In an early discovery call, it may signal low urgency. In a procurement email, it may indicate budget timing. In a post-demo conversation, it may suggest deal slippage. The same sentence has different operational meaning depending on context type, source, timing, and deal state.

Raw context further hides provenance. A statement extracted from a signed agreement, an email from a

decision-maker, a CRM note written by a salesperson, a support ticket, and an AI-generated summary should not be treated equally. Business decisions often depend not only on what was said, but also on who said it, when it was said, whether it was confirmed, and whether it can be verified. Without provenance, the model may over-weight weak signals and under-weight authoritative ones.

Another limitation is that raw context is poorly suited for transaction-level reasoning. Aggregate business analytics can use averages, distributions, conversion rates, and historical patterns to describe what usually happens across many deals or customers. However, an active business transaction requires reasoning about the specific person, account, timeline, and situation involved. A million-dollar deal cannot be treated as interchangeable with previous million-dollar deals simply because it belongs to the same revenue band. Statistical patterns are useful as priors, but the final recommendation must be grounded in the current transaction's specific context.

This distinction is critical because business outcomes are affected by partially observable and uncontrollable variables: buyer mood, timing, internal politics, seasonality, market pressure, urgency, competing priorities, and recent interactions. These factors make business workflows inherently uncertain. Therefore, the goal of an enterprise AI system should not be to produce deterministic claims from raw historical data, but to produce grounded recommendations based on the best available case-specific context, with clear assumptions and source traceability.

Raw context is therefore not enough because it is not automatically relevant, structured, recent, reliable, or decision-ready. The goal should not be to pass raw context into the model. The goal should be to convert raw context into usable business context: compact, source-aware, temporally relevant, permission-aware, and directly aligned with the task being performed. This is the role of the Context Layer.

5. DEFINING THE CONTEXT LAYER

The *Context Layer* is a middleware layer that transforms raw business data into structured, source-aware, task-relevant context for stateless LLM inference. It sits between operational data systems and the model call. Its purpose is not merely to retrieve information, but to decide what information should be made available to the model, in what form, and with what source grounding.

Modern context engineering treats context as a dynamically assembled set of informational components rather than a single static prompt [4]. Building on this view, the Context Layer proposed in this paper is designed specifically for business workflows where context is fragmented, time-sensitive, source-dependent, and transaction-specific.

Let a task-specific inference request be represented as τ, and let available business sources be represented as

$$S = \{s_1, s_2, \ldots, s_n\}. \tag{1}$$

Each source s_i may contain one or more context items. A context item x_i is represented as a tuple:

$$x_i = (c_i, src_i, t_i, a_i, r_i, type_i, perm_i, tok_i), \tag{2}$$

where c_i is the content, src_i is the source type, t_i is the timestamp, a_i is the actor or participant, r_i is the source reliability, $type_i$ is the business meaning, $perm_i$ indicates whether the item is permitted for use, and tok_i is the token cost of including the item.

The Context Layer applies a set of functions F over available sources to produce a compact context payload C_τ:

$$C_\tau = A\left(Select\left(Compress\left(Classify\left(Retrieve(S, \tau)\right)\right)\right)\right), \tag{3}$$

where $Retrieve$ identifies candidate information, $Classify$ assigns business meaning and source metadata, $Compress$ removes redundant or low-value content, $Select$ chooses the final items under budget

constraints, and A assembles the model-ready context payload. The LLM then produces an output Y conditioned on the task and assembled context:

$$Y \sim P_\theta(Y \mid \tau, C_\tau). \tag{4}$$

The Context Layer is not simply a vector database, memory system, summarizer, or prompt template. It is an orchestration layer that decides what the model needs to know for a specific inference call.

5.1. Source and Delivery as Two Separate Problems

The Context Layer has two distinct responsibilities. The first is the *source problem*: where does the context come from? The second is the *delivery problem*: how should that context be supplied to the LLM?

The source problem concerns the origin, freshness, permission status, and reliability of the data. Business context may come from PDFs, email threads, call transcripts, CRM records, product events, payment records, internal notes, support tickets, or previous AI-generated summaries. These sources should not be treated equally. A signed contract, a payment event, and an email from a decision-maker are generally stronger evidence than a salesperson's informal note or an automatically generated summary.

The delivery problem concerns the final form in which context is passed into the inference call. Raw text is rarely the ideal format. The Context Layer may deliver context as extracted clauses, structured fields, source-linked snippets, ranked timeline events, unresolved commitments, recent objections, or a compact context card. The same source can therefore produce different context payloads depending on the task.

5.2. Task-Conditional Noise

Noise in business context is not absolute. It is task-conditional. A sentence that appears irrelevant to a human may still help the model infer intent, urgency, sentiment, or relationship state. Conversely, a sentence that appears useful to a user may be redundant or distracting for the model if it repeats information already present in a more reliable source.

Therefore, this paper defines context noise as any information that consumes context budget without improving the expected quality of the task output. Formally, for task τ, an item x_i is noisy when its expected contribution to task reward is lower than its cost of inclusion:

$$Noise_\tau(x_i) = \mathbb{I}\left[\Delta Reward_\tau(x_i) < \lambda \cdot tok_i\right], \tag{5}$$

where tok_i is the token cost of including the item and λ controls the cost sensitivity of the system. This definition implies that noise must be evaluated relative to the task, model, and context budget.

In practice, redundant greetings, repeated follow-ups, outdated attachments, filler speech, duplicated CRM notes, and low-information acknowledgements often behave as noise. However, contradictions, unresolved commitments, objections, and changes in buyer intent should not be removed merely because they are old or inconsistent. These items may carry high business value.

5.3. Recency and Semantic Weighting

Business context changes over time. A recent buyer message often deserves higher priority than an older message because requirements, budgets, deadlines, and intentions evolve. However, recency alone is insufficient. A recent message such as "Okay, I will come" may have low business value, while an older message such as "We accept the proposal subject to legal approval" may remain highly important for months.

The Context Layer therefore assigns each context item a combined utility score. For a task τ, the utility of item x_i is defined as:

$$\begin{aligned} U_\tau(x_i) =& \alpha R_\tau(x_i) + \beta M_\tau(x_i) + \gamma Q(x_i) + \delta P_\tau(x_i) \\ &+ \eta D(x_i) - \zeta N_\tau(x_i), \end{aligned} \tag{6}$$

where $R_\tau(x_i)$ is task relevance, $M_\tau(x_i)$ is semantic business importance, $Q(x_i)$ is source reliability, $P_\tau(x_i)$ is persistence value, $D(x_i)$ is temporal recency, and $N_\tau(x_i)$ is redundancy or noise. The coefficients $\alpha, \beta, \gamma, \delta, \eta, \zeta$ can be tuned according to business domain, model behavior, and evaluation results.

Temporal recency can be represented using a decay function:

$$D(x_i) = \exp\left(-\frac{\Delta t_i}{h_{type_i}}\right), \tag{7}$$

where Δt_i is the age of the item and h_{type_i} is a type-specific half-life. For example, a casual acknowledgement may have a short half-life, while a signed contract clause, pricing commitment, or accepted proposal may have a long half-life.

Persistence value captures whether an older item remains active in the current business state. An unresolved objection, active commitment, accepted proposal, legal constraint, or pending approval should remain relevant even if it is not recent. This prevents the Context Layer from discarding old but business-critical information.

5.4. Context Cutoff as a Budgeted Selection Problem

The cutoff problem is central to context construction. A naive system may include all context above a fixed relevance threshold, such as 60% or 90%. However, this approach fails in business workflows because importance is not evenly distributed across time. A recent low-value message may not deserve inclusion, while an older high-value commitment may be essential.

Therefore, the Context Layer should treat cutoff as a constrained selection problem rather than a simple time-based or score-based filter. Given a candidate set I_τ, the final context payload is selected as:

$$C_\tau^* = \arg\max_{C \subseteq I_\tau} \left[\sum_{x_i \in C} U_\tau(x_i) - \rho \sum_{x_i, x_j \in C} Sim(x_i, x_j) \right], \tag{8}$$

subject to

$$\sum_{x_i \in C} tok_i \leq L_{eff}, \tag{9}$$

$$Cost(C) \leq B, \quad Latency(C) \leq T, \quad perm_i = 1 \; \forall x_i \in C. \tag{10}$$

Here, $Sim(x_i, x_j)$ penalizes redundancy between selected items, ρ controls the strength of redundancy removal, and L_{eff} is the effective context budget for the model and task. This formulation is similar in spirit to diversified retrieval approaches, where the goal is to select high-value but non-redundant evidence [19].

This cutoff method allows the Context Layer to preserve old but critical business anchors while still prioritizing recent and semantically meaningful updates. The final context is therefore not simply the latest context, nor the longest context, but the most useful context under operational constraints.

5.5. Context Layer Objective

The objective of the Context Layer is to maximize task reward under context-length, latency, cost, permission, and grounding constraints:

$$F^* = \arg\max_F \; \mathbb{E}_\tau \left[Reward(Y, Y_\tau^*) \right], \tag{11}$$

subject to

$$|C_\tau| \leq L_{max}, \quad Cost(C_\tau) \leq B, \quad Latency(C_\tau) \leq T. \tag{12}$$

In business workflows, this reward should not be measured only by output correctness. It should also include groundedness, source attribution, actionability, user acceptance, and downstream workflow performance. Prior work on context compression and long-context limitations supports the need to reduce redundant input while preserving task-relevant information [11], [12], [14].

Table 1. Business context source taxonomy for the Context Layer.

Source Type	Typical Content	Business Meaning	Context Handling Priority
PDF / document	Proposals, contracts, policies, brochures	Formal constraints, pricing, scope, terms	Extract clauses, requirements, and decision-relevant sections
Email	Buyer replies, commitments, approvals, objections	Intent, urgency, explicit commitments, decision stage	Prioritize recent decision-maker messages and unresolved commitments
Call transcript	Spoken objections, discovery notes, implicit intent	Needs, pain points, objections, sentiment	Remove transcript noise and extract business signals
CRM record	Deal stage, owner, value, lead status, notes	Operational state and structured history	Use as state anchor and validate against recent interactions
Transaction / event	Payment, demo booked, proposal sent, product action	Evidence of behavior or workflow progress	Treat as high-confidence factual state
Internal note	Sales comments, manager review, support insight	Human interpretation and operational judgment	Mark as interpretation, not primary evidence
AI summary	Prior summaries, extracted insights, generated notes	Compressed secondary representation	Use only with source links or confidence metadata

6. CONTEXT DENSITY

This paper introduces *context density* as a key measure for evaluating context quality. Context density refers to the amount of task-relevant, source-grounded information contained per token of context supplied to the model. In business workflows, the goal is not merely to increase the amount of context, but to increase the amount of useful business signal inside a limited context budget.

A low-density context contains large amounts of raw text with only a small percentage of useful information. A high-density context contains compact, structured, and relevant information that directly supports the task. For example, a low-density context for lead prioritization may include every email exchanged with a lead. A high-density context may include the lead stage, latest buyer intent, primary objection, last commitment, source of latest update, urgency level, and recommended next action constraint.

A simple operational definition is:

$$\text{Context Density} = \frac{\text{Relevant Source-Grounded Signals}}{\text{Total Context Tokens}}. \tag{13}$$

This definition captures the central trade-off in context construction. Every token supplied to the model has a cost. If a token contributes useful evidence, business state, source grounding, or task constraints, it increases density. If it repeats already-known information, introduces outdated context, hides the important signal, or adds irrelevant detail, it reduces density.

6.1. Density as Information Packing

Context density can be understood as the problem of packing the maximum amount of decision-relevant information into the smallest useful representation. This does not mean aggressive summarization alone. A summary may be short but still low-density if it removes source, time, actor, or business meaning. Similarly, a longer context may be high-density if every included item has a clear role in the decision.

For business workflows, useful context usually has five properties: it is relevant to the task, grounded in a source, temporally meaningful, semantically classified, and compact enough to fit within the effective model budget. Therefore, a stronger definition of context density for task τ can be written as:

$$CD_\tau(C) = \frac{\sum_{x_i \in C} U_\tau(x_i) \cdot G(x_i)}{Tokens(C)}, \tag{14}$$

where $U_\tau(x_i)$ is the task-specific utility of context item x_i, $G(x_i)$ is a grounding score representing source traceability and reliability, and $Tokens(C)$ is the total number of tokens in the assembled context. This formulation makes context density a weighted measure rather than a simple count of facts.

The Context Layer should therefore optimize for high-density context, not just short context. Removing too much information can damage reasoning, especially when the removed information contains exceptions, commitments, or causal explanations. The objective is selective compression: reduce redundancy while preserving business-critical signals [11].

6.2. Structured Context and Model Reasoning

The format of context strongly affects its density. Raw text often forces the model to infer structure implicitly. For example, if a prompt contains a long email thread, the model must identify who said what, when it was said, whether it is still valid, whether it is an objection or commitment, and whether it came from a reliable actor. This consumes reasoning capacity before the model can even solve the actual task.

Structured context reduces this burden by making business meaning explicit. JSON, CSV, tables, key-value records, and context cards can expose the schema of the problem directly to the model. Instead of asking the model to discover structure inside raw text, the Context Layer supplies structure as part of the input.

For example, the same information can be provided as raw text:

> The client said last week that they were interested but needed pricing approval. Yesterday, Priya asked for a revised quotation for 50 users and mentioned that legal approval would be needed before Friday.

Or as structured context:

```
{
"latest_intent": "requested revised quotation",
"quantity": "50 users",
"decision_constraint": "legal approval required",
"deadline": "Friday",
"actor": "Priya",
"source": "email",
"recency": "yesterday"
}
```

The structured version is denser because the business meaning is already labeled. The model does not need to infer which part is the actor, which part is the request, which part is the constraint, and which part is the timeline. This can improve reliability, especially in workflows where the model must compare many similar records.

6.3. Choosing the Right Context Format

No single structure is best for every task. The Context Layer should choose the format based on the type of information and the reasoning required.

Table 2. Context format choices and their role in context density.

Format	Best Use	Density Advantage	Risk
Raw text	Nuanced conversations, explanations, emotional tone	Preserves original wording and subtle meaning	Low density, noisy, hard to compare across records JSON
Nested facts, provenance, task-specific context cards	Explicit labels, source metadata, easy grounding	Can become verbose if overused CSV	Many repeated records with same fields
Very compact for rows of similar data	Weak for nested meaning and long explanations Table	Comparing entities, leads, deals, tasks, or events	Easy scanning, high compression, strong relational clarity
May lose nuance if fields are too short Key-value list	Small context cards and decision summaries	Compact and readable for inference	Can oversimplify complex evidence Timeline
Events where recency and sequence matter	Preserves order and state transitions	Can become long if not filtered	

For repeated business records such as leads, deals, tickets, product events, or transaction logs, tables and CSV-like formats are often more token-efficient than prose. For nested business context involving source, actor, timestamp, reliability, and evidence, JSON is often more expressive. For conversational nuance, raw snippets may still be necessary. Therefore, high-density context is usually hybrid: a structured context card supported by selected verbatim evidence.

6.4. Order, Schema, and Attention

Context density is not only about compression. It is also about order. A dense context can still perform poorly if the most important information is placed in a confusing sequence. Long-context research shows that relevant information can be underused when buried inside large context blocks [12], [13], [14]. Therefore, the Context Layer should order context according to business utility.

A practical ordering strategy is:

1. current state and task objective,
2. highest-priority recent signals,
3. unresolved commitments and objections,
4. source-grounded evidence,
5. historical anchors that remain active,
6. assumptions, missing information, and constraints.

This order gives the model a decision frame before evidence is introduced. It prevents the model from treating all information as equally important. The schema acts as an interface between business memory and model inference.

6.5. Density Loss and Over-Compression

Increasing density does not mean removing everything that looks redundant. Some repeated information is useful because repetition across sources can increase confidence. For example, if a buyer mentions the same

objection in a call and later repeats it in an email, the repetition is not merely noise. It confirms persistence of the objection.

Therefore, the Context Layer must distinguish between redundant wording and redundant evidence. Redundant wording should be compressed. Redundant evidence may be preserved as a confidence signal. This distinction is important because business decisions often depend on whether a signal is isolated, repeated, contradicted, or confirmed.

The risk of over-compression can be represented as density loss:

$$DL_\tau = 1 - \frac{\sum_{x_i \in C_{compressed}} U_\tau(x_i)}{\sum_{x_i \in C_{raw}} U_\tau(x_i)}. \tag{15}$$

A good Context Layer should reduce token count while keeping DL_τ low. In other words, it should remove low-utility tokens, not high-utility business signals.

6.6. Context Density as a Design Goal

The central design goal of the Context Layer is to increase context density while preserving the facts necessary for task completion. This means transforming raw business data into compact, structured, and source-aware payloads. The final payload should not be a generic summary. It should be a task-specific representation of the business state.

For example, for lead prioritization, the context card should expose lead stage, latest intent, urgency, deal value, decision-maker status, last objection, next promised action, and source confidence. For deal-risk detection, it should expose stalled timelines, unresolved objections, missing stakeholders, negative sentiment, delayed responses, and contradiction between CRM state and recent communication.

In this sense, context density is not only a token-efficiency metric. It is a measure of how well a system converts fragmented business memory into model-usable intelligence.

7. METHODOLOGY

This study proposes an experimental framework to evaluate whether source-aware, compressed, and structured context improves stateless LLM inference compared to raw long-context injection. The methodology is designed to test the central claim of this paper: business AI performance depends not only on model capability, but on the density, structure, provenance, recency, and task relevance of the context supplied during inference.

The evaluation compares multiple context construction strategies while keeping the base model, task prompt, temperature setting, and task distribution constant. This isolates the effect of context design from the effect of model capability.

7.1. Research Questions

The experiment is organized around six research questions:

1. Does source-aware compressed context improve business decision quality compared to raw full context?
2. Does structured context, such as JSON, tables, key-value records, or context cards, improve reasoning compared to unstructured raw text?
3. Does context density correlate with higher groundedness, lower hallucination, and better actionability?
4. Does recency-only selection underperform utility-based selection that combines recency, semantic importance, persistence, source reliability, and redundancy?
5. How much token cost and latency can be reduced without losing business-critical information?
6. How robust is each context condition when noisy, outdated, contradictory, or low-value context is added?

These questions evaluate the Context Layer not as a single retrieval component, but as a full context planning and assembly system.

7.2. Business Tasks

The evaluation should use enterprise tasks where output quality depends heavily on contextual business state. Candidate tasks include:

1. **Lead prioritization:** deciding which lead should be contacted first and why.
2. **Next-best-action generation:** recommending whether to call, email, wait, escalate, send pricing, or involve another stakeholder.
3. **Deal-risk detection:** identifying whether a deal is likely to stall, drop, or require intervention.
4. **Offer matching:** selecting the most relevant product, service, plan, or proposal angle for a buyer.
5. **Follow-up drafting:** generating a message grounded in the latest buyer intent, objection, and promised action.
6. **Conversation summarization:** producing a compact source-grounded summary of a business thread.
7. **CRM update recommendation:** suggesting updates to deal stage, lead status, next follow-up date, or risk notes.

These tasks cover both analytical outputs and action-oriented outputs. This distinction is important because business AI systems are not only expected to summarize information, but also to recommend and execute the next operational step.

7.3. Dataset Construction

The experiment can be performed using anonymized real business data, synthetic business workflows, or a hybrid dataset. Each workflow instance should contain multiple source types, such as email threads, call transcripts, CRM records, proposal documents, internal notes, and event logs.

Each instance should include:

1. a business task τ,
2. raw source documents and messages,
3. timestamps for each context item,
4. source type and actor metadata,
5. human-labeled business signals,
6. expected output or evaluator rubric,
7. token count and latency measurements for each context condition.

The dataset should intentionally include difficult cases: outdated information, contradictory signals, repeated objections, stale CRM fields, low-information recent messages, old but still-active commitments, and noisy transcripts. These cases are necessary to test whether the Context Layer can distinguish between latest information and most important information.

7.4. Context Conditions

Each task is tested across multiple context conditions while keeping the model constant. These conditions compare unstructured, retrieved, summarized, structured, and source-aware context strategies.

Table 3. Context conditions for experimental comparison.

Condition	Description
No context	Only the user request and system instruction are provided. This establishes the model's baseline behavior without business memory. Raw full context
Full PDFs, email threads, call transcripts, CRM notes, and other records are passed directly with minimal filtering. Raw newest-first context	Raw context is ordered by recency, with latest messages placed first. This tests whether recency alone improves performance. Retrieved snippets
Top-k semantically similar chunks are retrieved and appended using a retrieval system. Generic summary	A short summary is generated without source-aware classification, provenance tracking, or business signal labeling. Structured table / CSV
Repeated business records are supplied in compact rows and columns, preserving fields such as actor, timestamp, source, signal, and status. JSON context card	Context is supplied as a structured JSON-like payload with explicit fields for state, intent, objection, commitment, source, recency, and confidence. Source-aware context card
Classified, compressed, structured, source-grounded context is supplied with evidence references and business meaning labels. Utility-weighted context card	Context is selected using a utility function combining recency, semantic importance, source reliability, persistence, redundancy, and token cost. Noisy context
Irrelevant, outdated, duplicated, or contradictory context is added to test robustness.	

This comparison is designed to test whether the advantage comes from compression alone, structure alone, source awareness alone, or the combined Context Layer approach.

7.5. Context Density Measurement

For each context condition, context density is measured as the amount of task-relevant, source-grounded business signal per token:

$$CD_\tau(C) = \frac{\sum_{x_i \in C} U_\tau(x_i) \cdot G(x_i)}{Tokens(C)}, \tag{16}$$

where $U_\tau(x_i)$ is the task-specific utility of context item x_i, $G(x_i)$ is the grounding score based on source traceability and reliability, and $Tokens(C)$ is the total number of tokens in the context payload.

The utility score can be estimated through human labels, task-specific signal extraction, or evaluator

scoring. For example, a context item that contains the latest buyer objection, a signed pricing commitment, or a pending legal approval should receive higher utility than a greeting, repeated confirmation, or outdated note.

7.6. Context Selection and Cutoff Evaluation

To evaluate the cutoff mechanism, the experiment compares three selection strategies:

1. **Time-based cutoff:** include only context from the most recent time window.
2. **Score-based cutoff:** include all items above a fixed relevance threshold.
3. **Budgeted utility selection:** select context items that maximize utility under token, cost, latency, and permission constraints.

The budgeted utility selection is defined as:

$$C_\tau^* = \arg\max_{C \subseteq I_\tau} \left[\sum_{x_i \in C} U_\tau(x_i) - \rho \sum_{x_i, x_j \in C} Sim(x_i, x_j) \right]. \tag{17}$$

subject to

$$\sum_{x_i \in C} tok_i \leq L_{eff}. \tag{18}$$

where $U_\tau(x_i)$ is the utility of item x_i, $Sim(x_i, x_j)$ penalizes redundant context, ρ controls redundancy removal, and L_{eff} is the effective context budget. This evaluates whether the Context Layer can preserve old but business-critical anchors while excluding recent but low-value messages.

7.7. Evaluation Design

For each task, outputs should be judged across human and automated criteria. Human evaluators should score correctness, business relevance, actionability, and risk awareness. Automated evaluators can measure source attribution, hallucination rate, token cost, latency, consistency, and citation coverage.

A fair comparison should use:

1. the same base model across all context conditions,
2. the same system instruction and task prompt,
3. a low-temperature setting for stable comparison,
4. repeated runs to measure output variance,
5. blind human evaluation where reviewers do not know which context condition produced the output,
6. source-level labels to verify whether claims are grounded.

Because business outcomes are affected by external variables, evaluation should not rely only on downstream conversion or closure rates. Instead, the experiment should evaluate intermediate quality signals such as whether the output is grounded, useful, correctly sourced, timely, and aligned with the current business state.

8. RESULTS FRAMEWORK

This paper is currently positioned as a framework and research proposal. Therefore, this section defines the expected result patterns and the metrics required to validate the Context Layer empirically.

The expected result is that source-aware, structured, and utility-weighted context cards will outperform raw context injection on most business tasks. The expected gains are not limited to task accuracy. They should

also appear in groundedness, context density, token efficiency, latency, consistency, source attribution, and actionability.

8.1. Expected Result Pattern

Raw full context may perform well when the relevant answer is explicitly stated and the context is short. However, as source variety, redundancy, contradiction, and noise increase, raw context is expected to degrade. This is because the model must spend part of its reasoning capacity identifying structure, resolving source reliability, filtering outdated information, and finding the relevant signal inside a large context block.

Retrieved snippets are expected to improve token cost compared to raw full context, but may miss business state if retrieval is based only on semantic similarity. For example, a semantically similar email may not be the most recent, most reliable, or most operationally important source.

Generic summaries are expected to reduce length but may erase provenance, uncertainty, and important exceptions. This can lead to confident but weakly grounded outputs.

Structured formats such as tables, CSV, JSON, and key-value context cards are expected to improve consistency and reduce interpretation burden because they expose the schema of the business problem directly to the model. However, structure alone may not be sufficient if the selected information is stale, incomplete, or low-utility.

Source-aware context cards are expected to perform best when tasks require reasoning across recency, source reliability, commitments, objections, deal state, and next action. Utility-weighted context cards are expected to perform especially well in difficult cases where recent low-value messages compete with older high-value commitments.

8.2. Evaluation Metrics

8.3. Ablation Study

To understand which part of the Context Layer contributes most to performance, the experiment should include ablation studies. Each ablation removes one component while keeping the rest of the system constant.

8.4. Expected Findings

The expected findings are as follows:

1. **Context density should correlate with output quality.** Higher-density context should produce more grounded, actionable, and consistent outputs.
2. **Structured context should outperform raw context for operational tasks.** Tables, JSON, and context cards should help the model reason over actors, timestamps, sources, commitments, and constraints.
3. **Raw long context should degrade under noise.** As irrelevant, duplicated, or outdated context increases, raw full context should show higher hallucination and lower consistency.
4. **Recency alone should be insufficient.** Newest-first ordering should help in some cases, but should fail when older high-value commitments remain active.
5. **Utility-weighted selection should improve cutoff decisions.** Context selected by utility, persistence, source reliability, and redundancy control should outperform fixed time-window or fixed-threshold cutoffs.
6. **Source-aware context should improve trust.** Outputs with source metadata should produce better attribution, lower unsupported claims, and higher evaluator confidence.
7. **Compression should reduce cost and latency.** Selective compression should lower token usage and response time while preserving business-critical facts.

The ideal result is not that the Context Layer always produces shorter prompts. The ideal result is that it produces better prompts: prompts with higher business signal per token, clearer source grounding, stronger

Table 4. Suggested evaluation metrics for Context Layer experiments.

Metric	Question Answered	Measurement Approach	Expected Improvement
Task accuracy			
Did the model produce the correct business decision?	Human labels, expert review, or ground-truth comparison	Higher with source-aware and utility-weighted context Business relevance	Does the output address the actual business situation?
Human score from 1–5	Higher with structured context cards Actionability	Can a sales, support, or operations user act on the output?	Human score from 1–5
Higher with task-specific context cards Groundedness	Is the output supported by supplied evidence?	Source-match scoring and evaluator review	Higher with provenance metadata Source attribution
Are claims linked to the correct source?	Claim-to-source verification	Higher with source-aware context Hallucination rate	Did the model invent unsupported facts?
Unsupported-claim detection	Lower with structured evidence Context density	How much useful signal is supplied per token?	Utility-weighted signal score divided by token count
Higher after compression and structuring Token cost	How many tokens were used?	Prompt and completion token count	Lower after selective compression Latency
How long did inference take?	End-to-end response time	Lower with smaller payloads Consistency	Does the answer remain stable across repeated runs?
Output variance across repeated calls	Higher with structured context Recency handling	Does the model correctly prioritize recent changes?	Test cases with changing timelines
Higher with temporal metadata Persistence handling	Does the model preserve old but still-active commitments?	Test cases with old high-value facts	Higher with utility-weighted selection Noise robustness
Does irrelevant or outdated context degrade performance?	Performance drop under noisy context	Smaller drop with Context Layer Format efficiency	Which representation gives the best quality per token?
Compare raw text, table, JSON, and context card	Higher with task-appropriate structure		

Table 5. Ablation tests for Context Layer components.

Ablation	Purpose
Remove source metadata	
Tests whether provenance improves groundedness and reduces hallucination. Remove recency weighting	Tests whether temporal metadata improves handling of recent changes. Remove persistence scoring
Tests whether older commitments and unresolved objections are incorrectly discarded. Remove semantic classification	Tests whether labels such as objection, commitment, intent, and risk improve reasoning. Remove structured formatting
Tests whether raw text performs worse than tables, JSON, or context cards. Remove compression	Tests whether uncompressed context increases cost, latency, and confusion. Remove redundancy penalty
Tests whether repeated low-value context reduces context density.	

temporal ordering, and fewer irrelevant distractions.

8.5. Interpretation of Results

If source-aware context cards outperform raw full context, the result would support the central thesis that enterprise AI performance is a context architecture problem, not merely a model scaling problem. If structured formats outperform raw text, it would support the claim that context delivery format affects reasoning quality. If utility-weighted cutoff outperforms recency-only cutoff, it would support the idea that business context must preserve persistent commitments and not merely recent messages.

If the results show only minor differences between conditions, that would suggest either that the selected tasks are not context-sensitive enough, the model is already robust to raw context, or the context card design is not sufficiently informative. In that case, future work should test harder business workflows with more source variety, contradiction, noise, and time-dependent state changes.

9. DISCUSSION

The Context Layer reframes enterprise AI performance as a systems problem rather than only a model problem. Larger models, longer context windows, better tool access, and more capable agents are valuable, but they do not eliminate the need for context architecture. Business reasoning requires source-sensitive, time-aware, task-specific, and permission-aware context assembly. The central challenge is not only whether information exists, but whether the right information is selected, compressed, structured, grounded, and supplied to the model at the moment of inference.

A key implication of this paper is that the boundary between model intelligence and system intelligence must be made explicit. The model provides general reasoning, language generation, classification ability, and pattern recognition. The surrounding system provides business memory, source access, current state, permissions, workflow constraints, and evaluation feedback. Enterprise AI systems fail when these two responsibilities are collapsed into a single prompt or a simple retrieval pipeline. A generalist LLM can reason over a business situation only if the surrounding system converts fragmented operational data into a model-usable representation.

This distinction is especially important because LLM inference is stateless at the model level. A model does not automatically know the current status of a deal, the latest objection from a buyer, the payment state of a customer, or the internal escalation history of a support issue. These are operational facts, not general knowledge. Training and fine-tuning can teach a model how to classify leads, write follow-ups, reason over sales workflows, or follow a company tone. However, they cannot reliably replace live transactional context. A model trained last week cannot know what a customer said yesterday unless that information is stored externally, retrieved, selected, and reintroduced during inference.

The Context Layer also clarifies the limits of tool access. Modern agents may be able to query databases, inspect files, call APIs, access CRMs, read documents, execute code, and interact with external systems. However, access to a source is not the same as understanding which source matters. Before an agent can use the right information, it must know what context exists, what source should be inspected, what query should be issued, which result should be trusted, and what information should be carried forward into the final inference call. Tool access expands reach; the Context Layer determines relevance. This is why protocols, memory systems, agent workflows, and long-running agents still require context planning, persistent state, source indexing, and evaluable feedback loops [4], [6], [8], [9], [10].

Another major contribution of the Context Layer is the separation of raw context from usable context. Raw business data is rarely ready for inference. Emails contain repeated follow-ups and outdated threads. Call transcripts contain filler speech, interruptions, and transcription errors. PDFs contain legal, pricing, and operational sections that may not be relevant to the current task. CRM records may be stale, incomplete, or manually updated. Internal notes may contain useful human judgment but may not be authoritative evidence. Therefore, business context must be transformed before it is supplied to the model. This transformation includes source identification, semantic classification, compression, ordering, grounding, and assembly.

The concept of context density provides a useful design principle for this transformation. Instead of

maximizing context length, enterprise AI systems should maximize useful business signal per token. This changes the design objective from ‘how much can the model read?” to ‘how much task-relevant, source-grounded information is contained in each token?” A dense context payload is not merely short; it is compact, structured, source-aware, temporally meaningful, and aligned with the task. Prior work on context compression and long-context limitations supports this view: long input can increase cost and latency, and relevant evidence may still be underused when buried inside large or poorly organized context blocks [11], [12], [13], [14].

Structured context is one practical method for improving context density. JSON, CSV, tables, key-value records, timelines, and context cards can reduce the model’s burden of discovering structure from raw text. A table can make repeated lead records easier to compare. A JSON context card can preserve nested information such as actor, timestamp, source, confidence, and business signal. A timeline can preserve sequence and state transitions. However, structure must be chosen carefully. JSON can become verbose, CSV may lose nuance, and raw text may still be necessary when exact wording, sentiment, or ambiguity matters. The Context Layer should therefore use hybrid representations: structured context for state and comparison, selected verbatim evidence for nuance, and source metadata for grounding [20], [21], [22].

The recency problem further demonstrates why context selection cannot be solved through naive retrieval. In business workflows, recent information is often more important because requirements, budgets, and timelines change. However, recency alone is insufficient. A recent message such as ‘Okay, I will come” may contain little business value, while an older message such as ‘We accept the proposal subject to legal approval” may remain critical for months. The Context Layer must therefore balance temporal recency with semantic importance, source reliability, persistence, and redundancy. This leads to a budgeted selection view of context construction, where the system must preserve old but active business anchors while excluding recent but low-value information.

This has direct implications for business reasoning. Aggregate analytics can describe what usually happens across many customers, deals, or support cases. Transaction-level reasoning must decide what should be done for this specific customer, in this specific situation, at this specific time. A sales executive cannot treat one million-dollar deal as identical to previous million-dollar deals simply because they share a revenue band. The current buyer’s intent, internal urgency, procurement constraints, stakeholder dynamics, mood, timing, and recent interactions matter. Statistical patterns are useful as priors, but the final recommendation must be grounded in case-specific context.

The Context Layer also helps address evaluation complexity. Business outcomes are not fully deterministic. A good recommendation may fail because of buyer budget, competitor action, organizational politics, timing, market conditions, or factors outside the AI system’s control. Similarly, a weak recommendation may still be followed by a successful outcome for unrelated reasons. Therefore, evaluating enterprise AI only through final outcomes such as conversion rate or closure rate can be misleading. The Context Layer should also be evaluated through intermediate metrics: groundedness, source attribution, hallucination rate, actionability, context density, consistency, latency, token cost, and human acceptance [23], [24], [25].

Governance is another important implication. Business context may include private, sensitive, contractual, financial, or regulated information. Passing large raw payloads into every inference call increases exposure, cost, and audit complexity. A Context Layer can reduce this risk by minimizing the payload, preserving source metadata, applying permission checks, and ensuring that only necessary context is included. In this sense, context compression is not only an efficiency technique; it is also a governance mechanism. The less unnecessary context a model receives, the lower the risk of exposing irrelevant or sensitive information.

The broader implication is that enterprise AI systems should be designed as context-aware systems, not merely model-calling systems. The future of business AI will likely depend on how well systems manage the interface between operational memory and stateless reasoning. The model may become more capable, but the system must still decide what the model should know, what it should ignore, what it should cite, and what it should not be allowed to access. The Context Layer provides a framework for making these decisions explicit.

10. LIMITATIONS AND FUTURE WORK

This paper presents the Context Layer as a conceptual and methodological framework. Its primary limitation is that it requires empirical validation across real enterprise datasets. While the proposed framework is grounded in current research on context engineering, context compression, structured data reasoning, long-context limitations, and agentic systems, the specific claims about business workflows must be tested across domains, organizations, and task types.

A first limitation concerns dataset availability. High-quality business workflow data is difficult to obtain because it often contains private customer information, sales records, contracts, emails, call transcripts, support tickets, payment records, and internal notes. Public benchmarks rarely capture this level of operational complexity. As a result, future research should construct anonymized or synthetic business-context benchmarks that preserve the structure of real workflows without exposing sensitive information. Such benchmarks should include multi-source context, changing timelines, contradictory information, stale records, source reliability differences, and old but persistent commitments.

A second limitation is domain variation. The optimal Context Layer design may differ across sales, customer support, finance, hiring, healthcare, legal operations, procurement, and internal knowledge management. A sales workflow may prioritize buyer intent, urgency, objections, and next action. A support workflow may prioritize severity, reproduction steps, product logs, escalation history, and service-level agreements. A finance workflow may prioritize approvals, invoices, payment status, compliance rules, and audit trails. Therefore, context taxonomies, utility weights, source reliability scores, and compression strategies may need to be customized by domain.

A third limitation is the difficulty of measuring utility. This paper defines context density in terms of task-specific utility and source grounding. However, estimating utility is non-trivial. Human annotators may disagree about which signals are important. LLM-based evaluators may inherit model biases. Outcome-based validation may be noisy because business outcomes are influenced by external variables. Future work should compare multiple utility estimation methods, including expert annotation, evaluator models, outcome correlation, user feedback, and process-level metrics.

A fourth limitation concerns the risk of over-compression. While high-density context is desirable, excessive compression can remove nuance, uncertainty, contradictions, or source-specific wording. In business workflows, these details may be critical. A buyer's exact phrasing may reveal hesitation. A repeated objection may indicate persistence. A contradiction between a CRM field and a recent email may be more informative than either source alone. Future work should study how much compression is safe for different task types and when selected verbatim evidence should be preserved alongside structured summaries.

A fifth limitation is that structured context is not universally superior. JSON, CSV, tables, timelines, and context cards can improve clarity and reduce reasoning burden, but they can also introduce schema bias. If the schema omits an important field, the model may ignore that dimension. If fields are too short, nuance may be lost. If JSON is too verbose, token efficiency may suffer. Future research should compare different context card formats across tasks and measure not only accuracy, but also context density, groundedness, latency, and human preference.

A sixth limitation concerns model dependence. Different LLMs may respond differently to the same context format. Some models may handle raw long context well, while others may benefit more from structured payloads. Some models may be better at reasoning over tables, while others may perform better with natural language summaries. Therefore, Context Layer evaluation should be repeated across model families, context window sizes, and inference settings. Future work should also test whether the same context card structure transfers across models or must be optimized for each model.

A seventh limitation concerns system integration. A production Context Layer must operate under real constraints: API rate limits, data permissions, retrieval latency, storage architecture, user roles, logging requirements, security policies, and error recovery. The theoretical selection objective must therefore be adapted to engineering constraints. Future work should explore scalable implementations of context indexing, permission-aware retrieval, incremental summarization, source reliability scoring, and real-time context card generation.

Future work should proceed in several directions. First, researchers should build benchmark datasets for business-context reasoning. These datasets should include tasks such as lead prioritization, next-best-action generation, deal-risk detection, offer matching, follow-up drafting, CRM update recommendation, and support escalation. Each task should include raw context, source metadata, expected outputs, and human evaluation rubrics.

Second, future work should compare context construction strategies. Experiments should test no context, raw full context, newest-first raw context, retrieved snippets, generic summaries, structured tables, JSON context cards, source-aware context cards, and utility-weighted context cards. Such comparisons would help determine whether gains come from retrieval, compression, structure, source awareness, recency weighting, or their combination.

Third, future work should study context cutoff methods. Time-based cutoff, fixed threshold cutoff, and budgeted utility selection should be compared under token, cost, latency, and permission constraints. This is especially important in business workflows because older commitments, unresolved objections, and legal constraints may remain active even when they are not recent.

Fourth, future work should develop better evaluation metrics for Context Layers. In addition to task accuracy, evaluation should include context density, density loss, source attribution, groundedness, hallucination rate, actionability, consistency, latency, token cost, noise robustness, recency handling, persistence handling, and human trust. Evaluation should also distinguish between analytical tasks and action-oriented tasks.

Fifth, future work should study context governance. A Context Layer should not only improve output quality but also reduce unnecessary data exposure. Future systems should measure how much sensitive or irrelevant information is excluded, whether permission constraints are respected, whether source metadata is preserved, and whether outputs can be audited.

Finally, future work should explore learning Context Layer policies from feedback. User corrections, accepted recommendations, rejected outputs, workflow outcomes, and source-level error analysis can all provide signals for improving context selection over time. However, such learning must be carefully governed to avoid reinforcing biased or noisy business behavior. The long-term research direction is an adaptive Context Layer that learns which context formats, sources, and utility weights work best for each workflow while preserving auditability and control.

11. CONCLUSION

The next generation of AI workflow systems will not be built only by increasing model size, context window length, or tool access. They will be built by improving the way context is sourced, selected, compressed, structured, grounded, and supplied to stateless inference calls.

This paper introduced the *Context Layer* as a system layer between raw business data and LLM inference. The Context Layer converts fragmented operational memory into compact, source-aware, task-specific context. It addresses a central limitation of business AI systems: the model can only reason over what is made available to it at inference time, and raw business data is rarely suitable for direct use.

The paper argued that business context is fragmented, time-sensitive, source-dependent, permission-constrained, and often too large or noisy to pass directly into a model. It also argued that tool access does not eliminate the context problem. Even when an agent can access databases, emails, CRMs, documents, and APIs, it must still decide what to inspect, what to trust, what to ignore, what to compress, and what to carry forward into the final model call.

To address this, the paper proposed source-aware context construction, context density, utility-weighted selection, task-conditional noise removal, and structured context delivery. These ideas shift the design goal from maximizing context length to maximizing useful business signal per token. A good context payload is not simply shorter; it is more relevant, more grounded, more structured, more current, and more aligned with the task.

The paper also emphasized that business reasoning cannot rely only on aggregate statistical patterns.

Aggregate analytics can describe what usually happens across many customers or deals. Transaction-level reasoning must decide what should happen in one specific situation, with one specific buyer, at one specific time. This requires grounded, case-specific context with source traceability and explicit assumptions.

The Context Layer therefore provides both a technical and operational framework for enterprise AI. Technically, it improves inference by reducing noise, preserving important signals, and structuring context for model reasoning. Operationally, it improves governance by minimizing unnecessary data exposure, preserving provenance, and making outputs more auditable.

For business AI, the fundamental question is no longer only how much the model can read. The more important question is: what does the model need to know right now, in what structure, under what constraints, and from which source should it know it? The Context Layer is proposed as the system responsible for answering that question.

12. REFERENCES

[1] T. B. Brown et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[2] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[3] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems*, 2020.

[4] L. Mei et al., "A survey of context engineering for large language models," *arXiv preprint arXiv:2507.13334*, 2025.

[5] Salesforce, *Salesforce's agentforce is here: Trusted, autonomous ai agents now available*, Salesforce Newsroom, Accessed: 2026-06-20, Oct. 2024. [Online]. Available: `https://www.salesforce.com/in/news/press-releases/2024/10/29/agentforce-general-availability-announcement/`.

[6] S. Mishra-Sharma and Anthropic Discovery Team, *Long-running claude for scientific computing*, Anthropic Research Blog, Accessed: 2026-06-20, Mar. 2026. [Online]. Available: `https://www.anthropic.com/research/long-running-Claude`.

[7] Y. Wang, X. Chen, X. Jin, M. Wang, and L. Yang, "Openclaw-rl: Train any agent simply by talking," *arXiv preprint arXiv:2603.10165*, 2026.

[8] LangChain, *Langgraph overview*, LangChain Documentation, Accessed: 2026-06-20, 2026. [Online]. Available: `https://docs.langchain.com/oss/python/langgraph/overview`.

[9] Anthropic, *Introducing the model context protocol*, Anthropic News, Accessed: 2026-06-20, Nov. 2024. [Online]. Available: `https://www.anthropic.com/news/model-context-protocol`.

[10] OpenAI, *Openai agents sdk: Sessions*, OpenAI Agents SDK Documentation, Accessed: 2026-06-20, 2026. [Online]. Available: `https://openai.github.io/openai-agents-python/sessions/`.

[11] Y. Li, B. Dong, C. Lin, and F. Guerin, "Compressing context to enhance inference efficiency of large language models," *arXiv preprint arXiv:2310.06201*, 2023.

[12] N. F. Liu et al., "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024.

[13] J. He et al., "Never lost in the middle: Improving large language models via attention strengthening question answering," *arXiv preprint arXiv:2311.09198*, 2023.

[14] C. An et al., "Why does the effective context length of llms fall short?" *arXiv preprint arXiv:2410.18745*, 2024.

[15] T. Ju, W. Sun, W. Du, X. Yuan, Z. Ren, and G. Liu, "How large language models encode context knowledge? a layer-wise probing study," *arXiv preprint arXiv:2402.16061*, 2024.

[16] S. Sia, D. Mueller, and K. Duh, *Where does in-context learning happen in large language models?* NeurIPS 2024, 2024.

[17] J. Jiang, Y. Dong, J. Zhou, and Z. Zhu, "From compression to expression: A layerwise analysis of in-context learning," *arXiv preprint arXiv:2505.17322*, 2025.

[18] L. Chen et al., "Clasp: In-context layer skip for self-speculative decoding," *arXiv preprint arXiv:2505.24196*, 2025.

[19] J. Carbonell and J. Goldstein, "The use of mmr, diversity-based reranking for reordering documents and producing summaries," in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998, pp. 335–336.

[20] J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, and J.-R. Wen, "Structgpt: A general framework for large language model to reason over structured data," in *Proceedings of the 2023 Conference on Empirical Methods in Natural*

Language Processing, Association for Computational Linguistics, 2023, pp. 9237–9251. DOI: 10.18653/v1/2023.emnlp-main.574.

[21] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 8413–8426. DOI: 10.18653/v1/2020.acl-main.745.

[22] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," *arXiv preprint arXiv:2004.02349*, 2020.

[23] J. Pearl, *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2009.

[24] D. B. Rubin, "Estimating causal effects of treatments in randomized and nonrandomized studies," *Journal of Educational Psychology*, vol. 66, no. 5, pp. 688–701, 1974.

[25] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: Survey and practical guide," *Data Mining and Knowledge Discovery*, vol. 18, no. 1, pp. 140–181, 2009.